Semantic Annotation of RESTful Services Using External Resources

Victor Saquicela, Luis. M. Vilches-Blázquez, and Óscar Corcho

Ontology Engineering Group, Departamento de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid, Spain
{vsaquicela,lmvilches,ocorcho}@fi.upm.es

Abstract. Since the advent of Web 2.0, RESTful services have become an increasing phenomenon. Currently, Semantic Web technologies are being integrated into Web 2.0 services for both to leverage each other strengths. The need to take advantage of data available in RESTful services in the scope of Semantic Web evidences the difficulties to cope with syntactic and semantic description of the services.

In this paper we present an approach to tackle the problem of automatic the semantic annotation of RESTful services using a cross-domain ontology, a semantic resource (DBpedia) and additional external resources (suggestion and synonyms services) to annotate the parameters of the RESTful services. We also present a preliminary evaluation that proves the feasibility of our approach and highlights that it is possible to carry out this semantic annotation with satisfactory results.

Keywords: RESTful service, semantic annotation.

1 Introduction

In recent years, since the advent of Web 2.0, RESTful services have become an increasing phenomenon. They also play an important role in the Semantic Web by providing data to semantic software agents, as can be seen in [10, 12]. This rapid growth of RESTful services available on the Internet and the fact that the majority of the existing service descriptions have no semantic annotations, makes it possible to think of semantic description activities for them.

However, using RESTful services still requires much human intervention since the majority of their description pages are usually given in the form of unstructured text in a Web page (HTML), which contains a list of the available operations, their URIs and parameters (also called attributes), expected output, error messages, and a set of examples. The description includes all the details needed for a developer to execute the service or use it in applications such as mashups [3, 20, 21].

Traditionally, service semantic annotation approaches have focused on defining formalisms to describe these services [1, 5, 8]. These approaches take into account the description page of a RESTful service to carry out their semantic annotation processes.

The vast majority of RESTful services being reasonably well documented with respect to the functionality, programmers have been encouraged to supply an HTML

F. Daniel and F.M. Facca (Eds.): ICWE 2010 Workshops, LNCS 6385, pp. 266–276, 2010.

page with their services. Likewise, many approaches have resolved basic problems of RESTful service semantic description, but all related processes with RESTful service annotation are manual. Different examples of them can be found in [2, 11]. This is one of the main challenges of RESTful services that needs to be addressed in order to provide automation of semantic annotation tasks and to be able to interoperate with others applications or services.

In this paper, we focus on two main challenges: (1) to provide syntactic descriptions of RESTful service that allow their automatic registration and invocation, and (2) to interpret and enrich the RESTful services' parameters, by means of their semantic annotation.

Our main contribution is the proposal of an automatic approach for the semantic annotation of RESTful services using diverse types of resources: a cross-domain ontology, DBpedia, and diverse external services, such as suggestion and synonyms services.

The remainder of this paper is structured as follows: Section 2 presents related work in the context of semantic annotation of Web services and RESTful services. Section 3 introduces our approach for the automatic annotation of RESTful services, including deriving their syntactic description and semantic annotation. Section 4 presents a brief experimentation of our system. Finally, Section 5 presents some conclusions of this paper and future work.

2 Related Work

Most research in the semantic annotation of RESTful services has focused on the definition of formal description languages for creating semantic annotations. The main proposed formalisms for describing these services are: the Web Application Description Language[1] (WADL) which describes RESTful services syntactically, MicroWSMO [3] which uses hREST (HTML for RESTful services) [3, 5], and SA-REST [2, 8] which uses SAWSDL [1] and RDFa[2] to describe service properties.

In [19] the authors introduced an approach to annotate WADL documents linking them to ontologies. Among these approaches, some authors propose rather heavy-weight approaches for semantic description, which are normally derived from Web Service (WS-*) semantic description frameworks like WSMO or OWL-S. An example is proposed in [10], which makes use of a specific selection of existing languages and protocols, reinforcing its feasibility. Firstly, OWL-S is used as the base ontology for services, whereas WADL is used for syntactically describing them. Secondly, the HTTP protocol is used for transferring messages, defining the action to be executed, and also defining the executing scope. Finally, URI identifiers are responsible for specifying the service interface. Nevertheless, these languages are strongly influenced by existing traditional Web Services.

Other approaches are more lightweight, for instance, the proposals of [1, 2]. The authors advocate an integrated lightweight approach for formally describing semantic RESTful services. This approach is based on use of the hREST and MicroWSMO

[1] http://www.w3.org/Submission/wadl/
[2] http://www.w3.org/TR/xhtml-rdfa-primer/

microformats, which enable the creation of machine-readable service descriptions and the addition of semantic annotations. Furthermore, the authors present SWEET, a tool which effectively supports users in creating semantic descriptions of RESTful services based on the aforementioned technologies.

Our work can be considered as an extension of the work presented in [13, 14], in which the development of domain-independent approaches to semantically label Web services is described. The authors propose to automatically learn the semantics of information sources labelling the input and output parameters used by the source with semantic types of the user's domain model. In our approach we have dealt with RESTful services and have used semantic repositories to try to semantically label these services.

Another approach is presented in [17]. This approach classifies data type using HTML treated Web form files as the Web service's parameters. They use Naïve Bayes to classify assigned semantic types to the input and output parameters.

Likewise, our work is also similar to the approaches related to schema matching or integration [18, 16]. In these proposals the main goal is to establish semantic mappings between two different schemas. In our approach the developed system sets matchings between different parameters of a RESTful service and the DBpedia ontology[3].

3 An Approach for Automatic Semantic Annotation of RESTful Services

In this section, we present our approach visualized in Figure 1 for automating the syntactic and semantic annotation of RESTful services. Our system consists of three main components, including invocation and registration, repository, and semantic annotation components, which are enriched by diverse external resources. Next, we briefly describe the different components, illustrating the descriptions with some sample services on the geographical domain.

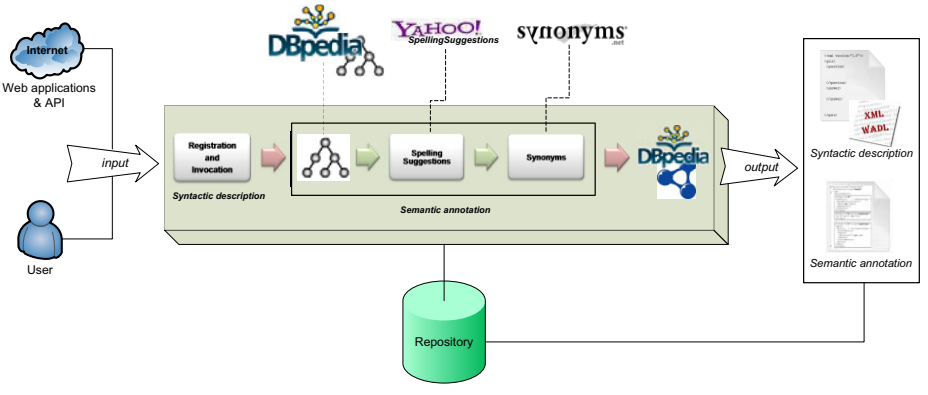

Fig. 1. RESTful Service Semantic Annotation System

[3] http://wiki.dbpedia.org/Ontology

3.1 A Sample Set of RESTful Services in the Geospatial Domain

The following services are two representatives of RESTful services in the geospatial domain taken from programmableweb.com:

- **Service 1.** http://ws.geonames.org/countryInfo?country=ES

This service retrieves information related to 'country', specifically, it returns information about the following parameters: 'capital', 'population', 'area' (km^2), and 'bounding box of mainland' (excluding offshore islands).

- **Service 2.** http://api.eventful.com/rest/venues/search?app_key=p4t8BFcLDtCzpxdS&location=Madrid

This service retrieves information about places (venues), specifically, it returns parameters like: 'city', 'venue_name', 'region_name', 'country_name', 'latitude', ' longitude', etc.

3.2 Syntactic Description: Invocation and Registration

Our system takes as input Web applications and APIs, which are known by users, or users can add manually a URL of an available RESTful service. In this case, we add manually different URLs of services and obtain automatically information related to each of the aforementioned RESTful service. Once URLs have been added, our system invokes the RESTful service with a sample of parameters and analyzes the response to obtain a basic syntactic description of a parameter set, which is used like inputs and outputs.

In this process our system uses the Service Data Object[4] (SDO) API to perform the invocation of the RESTful service and determine whether it is available or not. SDO is a specification for a programming model that unifies data programming across data source types and provides robust support for common application patterns in a disconnected way [22]. The invocation process is performed as follows: first, it takes the input parameters and their values, which are given to the service as part of a URL. Then, the system invokes the service which translates our "RESTful service call" into a query to specific service, including the URL and related parameters.

The service invocation of a specific RESTful service may return diverse formats, such as HTML, JSON, XML, etc. In our work we use only XML response for describing the service. The results of invocation of both services are showed in Table 1.

These XML responses are processed using SDO, which enables to navigate through the XML and extract output parameters of each service. The result of this invocation process is a syntactic definition of RESTful services in XML, which can be expressed in description languages like WADL or stored into a relational model. In this work we use a relational model as data model as a consequence of the simplicity of WADL for showing concepts. Table 2 shows the different output parameters of each service.

The output parameters are registered and stored into a repository. This repository is a database specifically designed to store syntactic descriptions of RESTful services. We selected this storage to increase efficiency in the recovery of the RESTful services.

[4] http://www.oasis-opencsa.org/sdo

Table 1. XML response of two sample RESTful services

Service 1	Service 2
<geonames> <country> <countryCode>ES</countryCode> **<countryName>Spain</countryName>** <isoNumeric>724</isoNumeric> <isoAlpha3>ESP</isoAlpha3> <fipsCode>SP</fipsCode> <continent>EU</continent> <capital>Madrid</capital> <areaInSqKm>504782.0</areaInSqKm> <population>40491000</population> <currencyCode>EUR</currencyCode> <languages>es-ES,ca,gl,eu</languages> <geonameId>2510769</geonameId> <bBoxWest>-18.169641494751</bBoxWest <bBoxNorth>43.791725</bBoxNorth> <bBoxEast>4.3153896</bBoxEast> <bBoxSouth>27.6388</bBoxSouth> </country> </geonames>	<venue id="V0-001-000154997-6"> <url>http://eventful.com/madrid/venues/la-ancha-/V0-001-000154997-6</url> **<country_name>Spain</country_name>** <name>La Ancha</name> <venue_name>La Ancha</venue_name> <description></description> <venue_type>Restaurant</venue_type> <address></address> <city_name>Madrid</city_name> <region_name></region_name> <region_abbr></region_abbr> <postal_code></postal_code> <country_abbr2>ES</country_abbr2> <country_abbr>ESP</country_abbr> <longitude>-3.68333</longitude> <latitude>40.4</latitude> <geocode_type>City Based GeoCodes</geocode_type> <owner>frankg</owner> <timezone></timezone> <created></created> <event_count>0</event_count> <trackback_count>0</trackback_count> <comment_count>0</comment_count> <link_count>0</link_count> <image></image> </venue> <venue id="V0-001-000154998-5">

Table 2. Syntactic description of RESTful service

Service 1:
countryInfo($country,bBoxSouth,isoNumeric,continent,fipsCode,areaInSqKm,languages, isoAlpha3,countryCode,bBoxNorth,population,bBoxWest,currencyCode,bBoxEast,capital, geonameId,countryName)
Service 2:
rest/venues/search($location,$app_key,id,link_count,page_count,longitude,trackback_count, version,venue_type,owner,url,country_name,event_count,total_items,city_name,address,name, latitude,page_number,postal_code,country_abbr,first_item,page_items,last_item,page_size, country_abbr2,comment_count,geocode_type,search_time,venue_name)

3.3 Semantic Annotation

Once the RESTful service is syntactically described with all its identified input and output parameters, we proceed into its semantic annotation. We follow a heuristic approach that combines a number of external services and semantic resources to propose annotations for the parameters as show in Figure 2. Next, we describe the main components of the semantic annotation.

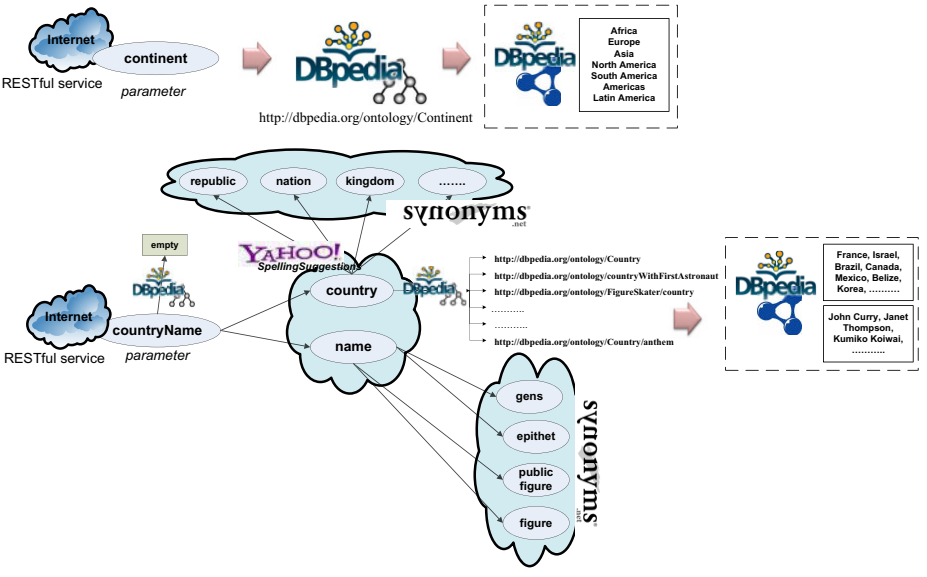

Fig. 2. Semantic annotation process

3.3.1 Using DBpedia in the Semantic Annotation

Currently, the RESTful service semantic annotation has some difficulties, which are briefly described in [1, 11]. In order to cope with them, we rely on techniques and processes that permit: a) semantic annotation only using syntactic description and, input/output parameters, or b) identification of some right example values that allow the invocation RESTful service automatically.

The starting point of the semantic annotation process is the list of syntactic parameters obtained previously. These parameters are used to query the DBpedia SPARQL Endpoint and retrieve the associated results to each parameter, as follows:

- First, the system retrieves all the classes from the DBpedia ontology whose names have an exact match with each parameter of the RESTful service. If the system obtains correspondences from the matching process, it uses these DBpedia concepts individually to retrieve samples (concept instances) from the DBpedia SPARQL Endpoint. The resulting information (RDF) is suggested automatically to the system and registered as a possible value for a certain parameter. When a parameter matches more than once in the DBpedia ontology, our system only considers concepts that have information (instances), and automatically discards those ontology concepts without instances.

  In order to retrieve information about identified parameters of RESTful services the system has registered the DBpedia SPARQL Endpoint as a service. This service enables automatically invocation of SPARQL queries over DBpedia Endpoint. Next, we present queries used by the system for retrieving DBpedia information.

Table 3. SPARQL query for retrieving classes of the DBpedia ontology (*filter omitted*)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-
ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#>
select distinct ?class where { ?class rdf:type
owl:Class. FILTER ..... }
```

This SPARQL query (see Table 3) enables to retrieve classes of the DBpedia ontology. The results of this query are compared to the concepts with each parameter of a service.

- Next, the system tries to find correspondences between parameters of the RESTful service and DBpedia properties. If the system obtains some correspondences, it uses these DBpedia properties individually to retrieve information of the DBpedia SPARQL Endpoint. Furthermore, this information is registered as a possible right value for a certain parameter.

  This SPARQL query (see Table 4) enables to retrieve properties of the DBpedia ontology. The system uses results to compare them with each parameter identified in the syntactic description.

Table 4. SPARQL query for retrieving properties of the DBpedia ontology (*filter omitted*)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-
ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#>  se-
lect distinct ?property  where {  ?property rdf:type
owl:ObjectProperty. FILTER .....}
```

- Finally, with the classes and properties matched, the system calls the DBpedia SPARQL Endpoint to retrieve values (instances) for classes and properties. Next we show some query examples:

Table 5. SPARQL query for retrieving possible values for a class

```
PREFIX  owl:  <http://www.w3.org/2002/07/owl#>  select
distinct ?val where { <" + c + "> a owl:Class. ?val a
<" + c + "> }
```

This SPARQL query (see Table 5) enables to retrieve possible values for a certain class of the ontology.

Table 6. SPARQL query for retrieving possible values for a property

```
PREFIX  owl:  <http://www.w3.org/2002/07/owl#>  select
distinct ?val where { <" + c + "> a owl:ObjectProperty.
?val <" + c + "> ?b}
```

This SPARQL query enables to retrieve possible values for a certain property of the ontology.

3.3.2 Enriching the Semantic Annotation

Since we request exact matches with DBpedia classes and properties, our system does not normally establish correspondences with ontology classes or properties for all parameters of the RESTful service. In order to annotate semantically the parameters that did not match any DBpedia resource, we add different external services to enrich the results. Below we describe the main characteristics of the external services added to the system.

Spelling Suggestion

Web search engines (e.g. Google, Yahoo, and Microsoft) usually try to detect and solve users' writing mistakes. The suggestions services, also called "Did You Mean", are spelling algorithms which solve these mistakes. For example, when a user writes 'countryName' these algorithms suggest 'country' and 'name' separately.

In our system we use the Yahoo Boss service[5] to retrieve suggestions about the parameters. Thus, for each parameter that the system did not find a correspondence with classes or properties, this service is invocated for obtaining a list of suggestions to query DBpedia again. The output is registered and stored into the repository. Following the previous example, the parameter 'countryName' is not found in the DBpedia ontology. Nevertheless, the added service allows separating this parameter in 'country' and 'name', and then it calls to the DBpedia SPARQL Endpoint for obtaining results.

Use of Synonyms

This external service[6] is incorporated to the system to retrieve synonyms of a certain parameter. This service tries to improve the semantic annotation process when our system does not offer results for the previous steps, that is, when we still have parameters in a RESTful service without annotations.

In the next example we find a parameter called 'address'. The invocation process uses the synonyms service to retrieve a set of synonyms of 'address' such as extension, reference, mention, citation, denotation, destination, source, cite, acknowledgment, and so on. These outputs are registered and stored into the repository, and then, the service calls to the DBpedia SPARQL Endpoint for results.

3.4 Checking the Semantic Annotation of RESTful Services

In order to check the collected semantic annotations of the previous process our system invocates the RESTful service, which was registered previously (as we describe in Section 3.2) with a random value of instances obtained from the queries to the DBpedia SPARQL Endpoint. If the system collects an instance value from the DBpedia SPARQL Endpoint and it is not empty, then our system considers that the invocation response is right based on the syntactic description. The system does not check all the collected instances as a default option, because there are many amount RESTful services with invocation limitations.

[5] http://developer.yahoo.com/search/boss/boss_guide/Spelling_Suggest.html
[6] http://www.synonyms.net/

The correspondences established between different parameters of a RESTful service and the DBpedia ontology (classes and properties) are registered and stored in the repository. In this way, the RESTful service is annotated semantically and it will allow generating semantic documentation of the type of service. An example of this can be seen in Table 7. This repository is a database specifically designed to store semantic annotations of RESTful services. As mentioned above, this storage is selected to increase efficiency in the recovery of RESTful services.

Table 7. Semantic annotation of a RESTful service

```
($country,bBoxSouth,isoNumeric,http://dbpedia.org/ontology/Con
tinent,fipsCode,http://dbpedia.org/property/areaMetroKm,langua
ges,isoAlpha3,http://dbpedia.org/ontology/country,bBoxNorth,ht
tp://dbpedia.org/ontology/populationDensity,bBoxWest,http://db
pedia.org/ontology/Currency,bBoxEast,http://dbpedia.org/ontolo
gy/capitalgeonameId,http://dbpedia.org/ontology/country)
```

4 Experimental Results

In order to evaluate our approach we use 12 different RESTful services founded in http://www.programmableweb.com/, which are characterized to contain geospatial information. The list of RESTful services can be seen in this website[7].

This analysis follows the three steps described in the semantic annotation. First, our system identifies correctly 16 parameters calling directly the DBpedia ontology but it fails to recognize 161 parameters. Second, the system uses the suggestion service and calls the DBpedia ontology. In this case, it identifies 41 correspondences, but it fails to recognize 120 parameters. Third, the system uses the synonyms service and calls the DBpedia ontology. It identifies 19 correspondences, but fails to recognize 101. A detailed view of these results is shown in Table 8.

Table 8. Results of the service test

RESTful service	Parameters	DBpedia ontology	Remaining parameters	Suggestions service	Remaining parameters	Synonyms service	Annotated parameters
*Source*1	17	**3**	14	**3**	11	**2**	8
*Source*2	24	**2**	22	**7**	15	**1**	10
*Source*3	7	**0**	7	**3**	4	**1**	4
*Source*4	13	**2**	11	**5**	6	**0**	7
*Source*5	14	**1**	13	**2**	11	**0**	3
*Source*6	11	**1**	10	**4**	6	**1**	6
*Source*7	7	**0**	7	**0**	7	**0**	0
*Source*8	8	**1**	7	**1**	6	**0**	2
*Source*9	43	**1**	42	**14**	28	**5**	20
*Source*10	4	**0**	4	**2**	2	**1**	3
*Source*11	7	**0**	7	**0**	7	**0**	0
*Source*12	22	**5**	17	**0**	17	**8**	13
Total	**177**	**16**	**161**	**41**	**120**	**19**	**76**

[7] http://castor.dia.fi.upm.es/ev/RESTfulservice.html

We cannot guarantee the success of the system in all the cases, because in some cases the system has not found any correspondence between RESTful service parameters and the concepts or properties of the DBpedia ontology.

After having analysed our findings, we have seen that some parameters are useless, because they refer to a navigation process through RESTful service results, for example: page, total, hits, etc. These parameters make it difficult to carry out the right annotation semantic process. We are planning to discard these types of parameters in the future.

To the best of our knowledge, there are no available results from existing research works to compare our results against. Likewise, these preliminary results prove the feasibility of our system and highlight that is possible to carry out an automatic semantic annotation of RESTful services.

5 Conclusions and Future Work

In this paper we have proposed an approach to perform an automatic semantic annotation process of RESTful services. This process is implemented in a system which takes into account the DBpedia ontology and its SPARQL Endpoint, as well as different external resources such as synonyms and suggestion services. We use combinations of these resources to discover meanings for each of the parameter of the RESTful services and perform semantic annotations of them.

In this work we use two different RESTful services related to the geospatial domain to guide the explanation of the proposed semantic annotation process. Finally, we have presented some preliminary experimental results that prove the feasibility of our approach and show that it is possible to carry out a semantic annotation of RESTful services automatically.

Future work will focus on the development of a GUI that will connect to services provided for the system. This online tool will be able to register and invoke new RESTful services, and annotate a lot of existing RESTful services with semantics. This can be useful for creating new applications (for instance, mashups) or for its use in the Semantic Web. Moreover, we also plan to make several improvements to the proposed system, related to the matching process and the use of similarity metrics. In the same sense, we also aim at improving in the SPARQL queries to DBpedia to better explore the knowledge of this resource in the annotation process, and optimize the use of suggestion and synonyms services.

Acknowledgments

This work has been supported by the R&D project España Virtual (CENIT2008-1030), funded by Centro Nacional de Información Geográfica and CDTI under the R&D programme Ingenio 2010. We would also like to thank Boris Villazón-Terrazas for his valuable comments.

References

1. Maleshkova, M., Kopecky, J., Pedrinaci, C.: Adapting SAWSDL for Semantic Annotations of RESTful Services. In: Workshop: Beyond SAWSDL at OnTheMove Federated Conferences & Workshops, Vilamoura, Portugal (2009)

2. Maleshkova, M., Pedrinaci, C., Domingue, J.: Semantically Annotating RESTful Services with SWEET. In: Demo at 8th International Semantic Web Conference, Washington D.C., USA (2009)
3. Maleshkova, M., Gridinoc, L., Pedrinaci, C., Domingue, J.: Supporting the Semi-Automatic Acquisition of Semantic RESTful Service Descriptions. Poster at ESWC 2009 (2009)
4. Pedrinaci, C., Domingue, J., Krummenache, R.: Services and the Web of Data: An Unexploited Symbiosis. In: Workshop: Linked AI: AAAI Spring Symposium "Linked Data Meets Artificial Intelligence" (2010)
5. Kopecký, J., Gomadam, K., Vitvar, T.: hRESTS: An HTML Microformat for Describing RESTful Web Services. In: Web Intelligence 2008, pp. 619–625 (2008)
6. Lambert, D., Domingue, J.: Grounding semantic web services with rules. In: Workshop: Semantic Web Applications and Perspectives, Rome, Italy (2008)
7. Steinmetz, N., Lausen, H., Brunner, M.: Web Service Search on Large Scale. In: ICSOC/ServiceWave 2009, pp. 437–444 (2009)
8. Lathem, J., Gomadam, K., Sheth, A.P.: SA-REST and (S)mashups: Adding Semantics to RESTful Services. In: ICSC 2007, pp. 469–476 (2007)
9. García Rodríguez, M., Álvarez, J.M., Berrueta, D., Polo, L.: Declarative Data Grounding Using a Mapping Language. Communications of SIWN 6, 132–138 (2009)
10. Freitas Ferreira Filho, O., Grigas Varella Ferreira, M. A.: Semantic Web Services: A RESTful Approach. In: IADIS International Conference WWW/INTERNET 2009, Rome, Italy (2009)
11. Alowisheq, A., Millard, D.E., Tiropanis, T.: EXPRESS: EXPressing REstful Semantic Services Using Domain Ontologies. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 941–948. Springer, Heidelberg (2009)
12. Alarcon, R., Wilde, E.: Linking Data from RESTful Services. In: Linked Data on the Web, LDOW 2010 (2010)
13. Lerman, K., Plangprasopchok, A., Knoblock, C.A.: Semantic Labeling of Online Information Sources. Int. J. Semantic Web Inf. Syst. 3(3), 36–56 (2007)
14. Ambite, J.L., Darbha, S., Goel, A., Knoblock, C.A., Lerman, K., Parundekar, R., Russ, T.A.: Automatically Constructing Semantic Web Services from Online Sources. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 17–32. Springer, Heidelberg (2009)
15. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In: SIGMOD Conference 2001, pp. 509–520 (2001)
16. Doan, A., Domingos, P., Halevy, A.Y.: Learning to Match the Schemas of Data Sources: A Multistrategy Approach. Machine Learning 50(3), 279–301 (2003)
17. Heß, A., Kushmerick, N.: Learning to Attach Semantic Metadata to Web Services. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 424–440. Springer, Heidelberg (2009)
18. Rahm, E., Bernstein, P.: On matching schemas automatically. VLDB. Journal 10(4) (2001)
19. Battle, R., Benson, E.: Brinding the semantic web and web 2.0 with Representational State Tranfer (REST). Web Semantics 6, 61–69 (2008)
20. Braga, D., Ceri, S., Martinenghi, D., Daniel, F.: Mashing Up Search Services. IEEE Internet Computing 12(5), 16–23 (2008)
21. Altinel, M., Brown, P., Cline, S., Kartha, R., Louie, E., Markl, V., Mau, L., Ng, Y.H., Simmen, D., Singh, A.: Damia: a data mashup fabric for intranet applications. In: VLDB 2007: Proceedings of the 33rd international conference on Very large data bases, pp. 1370–1373. VLDB Endowment (2007)
22. Resende, L.: Handling heterogeneous data sources in a SOA environment with service data objects (SDO). In: Proceedings of the ACM SIGMOD international conference on Management of data, pp. 895–897. ACM, New York (2007)